ISSUE #008 · BORROWED TRUST

Borrowed Trust.

A single GitHub issue, from a stranger with no write access, turns Anthropic's official Claude Code Action into code execution and a repo-scoped token. A command channel rides out through an on.aws Lambda URL that reads like every other AWS call. And a policy layer now catches a coding agent's tool calls before they run. Same thread through all three: attackers walked in on a primitive you already trust.

> ATTACK

RyotaK at GMO Flatt Security · 2026-06-01 · **Poisoning Claude Code: One GitHub Issue to Break the Supply Chain**

Plenty of repos wire Anthropic's official `claude-code-action` into issue triage: tag the bot, Claude reads the issue, Claude replies. RyotaK turned that into remote code execution from the outside. He plants fake error-message text in the issue body, and when the action calls `mcp__github__get_issue` to read it, that text steers Claude into running attacker commands (it took extra prompting to make reliable). Indirect prompt injection, with a privileged CI runner as the sink. In agent mode the `checkWritePermissions` guard meant to block untrusted actors waved through any GitHub App regardless of its permissions, and a GitHub App can open an issue on a public repo without being installed.

From execution on the runner, the payload reads `/proc/self/environ` and lifts the OIDC request token and URL (`ACTIONS_ID_TOKEN_REQUEST_TOKEN` and `..._URL`), then ships them out through `mcp__github__update_issue`. With those, the attacker runs the OIDC exchange and mints the Claude GitHub App installation token, scoped to the target repo with write to contents, issues, pull requests, and workflows. **One issue from an actor with zero write access reaches code-push on that repo; if the target is an action others depend on, the write propagates downstream. The action you trust to triage issues is the entry point, because it reads attacker text on a privileged runner.** Around 50 separate vulnerabilities, fixed in `@v1.0.94`. CVSS 7.8. RyotaK reported it rather than pushing to Anthropic's repos.

▸ SHIP THIS WEEK

Pin `claude-code-action` to `@v1.0.94` or later, where the trigger now runs a `checkHumanActor` check that drops GitHub App actors. Strip `issues: write`, `contents: write`, and `id-token: write` from any workflow an untrusted issue or PR can trigger; an AI action that reads issue text is handling attacker-controlled input. Then grep your workflows for `allowed_non_write_users: "*"`, the wildcard that opens the door.

> DEFENDER

Leonardo Grasso at Falco · 2026-05-12 · **Introducing Prempti: Falco meets AI coding agents**

Prempti puts Falco rules between Claude Code and execution. The agent declares a tool call, Prempti evaluates it first, and back comes allow, deny, or ask. Default rules cover reads of `~/.ssh`, `~/.aws`, and `.env`, pipe-to-shell, IMDS access, reverse shells, MCP config poisoning, and git-hook persistence. Rules match on fields like `tool.input_command` and `agent.cwd`, so denying any call where `agent.cwd` sits outside your project root takes a few lines in `~/.prempti/rules/user/`. **Know the honest limit: Prempti reads the tool call the agent declares, and the OS-level behavior it produces stays out of view.** Run Monitor mode for a week, then flip to Guardrails.

> AGENT BENCH

The coding agent you'd point at a poisoned dependency reads your creds and runs your shell, which makes it a trusted primitive too. Hand it the first audit pass. Scope one package and exact version, and ask for install and build scripts (`preinstall`, `postinstall`, `build.rs`) that touch the network or read `~/.aws`, `~/.ssh`, or `.env`, plus a diff between the published `dist/` and the source tag. Require a source-to-sink trace per hit. Then run one check deterministically: grep your repos and agent config files (`.cursorrules`, `CLAUDE.md`, `.claude/`) for zero-width Unicode, the trick the TrapDoor campaign used to smuggle instructions past you. **Out comes a ranked list at `file:line` of dependencies whose install path can reach your secrets. Agents over-report, so your verification does the gating. This is the oss-security-audit workflow.**

> RADAR

Aniket Harne at Qualys · 2026-06-02 · **The HazyBeacon Protocol: How Malware Weaponizes AWS Lambda Function URLs**

A re-dissection of the 2025 HazyBeacon campaign, kept here for one mechanic. The implant beacons to a Lambda Function URL set to `AuthType: NONE`. Because that endpoint lives on an `on.aws` domain, the channel inherits AWS's reputation and reads as routine encrypted HTTPS. A relay Lambda logs the metadata and forwards the payload to a backend. Harne's defender moves: an SCP that blocks creating Function URLs with `AuthType: NONE` unless policy-approved, org-wide CloudTrail, and a watch on Lambda cost anomalies. C2 hides best inside the domains you already trust.

> RECON ROLES

Tailscale, Security Infrastructure Engineer (Product Security). Remote, two listings: United States ($163,000–$226,000 USD) and Canada (CA$218,420–CA$302,840). Cloud-platform security with an AWS lean, Go preferred, plus Kubernetes, CI/CD, and infrastructure audits that drive remediation. Hands-on infra-and-codebase work that suits an engineer who codes. Tailscale posted both US and Canada with public ranges in both currencies, so for once you can compare an opening straight across the border.

SOURCES

GMO Flatt Security / flatt.tech/research (Poisoning Claude Code, RyotaK)
Falco / falco.org/blog/introducing-prempti
Qualys / blog.qualys.com (HazyBeacon Protocol, Aniket Harne)
Tailscale / job-boards.greenhouse.io/tailscale (Security Infrastructure Engineer, US + CA)